PART I

Install, Configure, and Manage the vSphere Environment

COPYRIGHTED MATERIAL

- CHAPTER 1: AUTOMATING VCENTER SERVER DEPLOYMENT AND CONFIGURATION
- CHAPTER 2: AUTOMATING VSPHERE HYPERVISOR DEPLOYMENT AND CONFIGURATION
- CHAPTER 3: AUTOMATING STORAGE AND NETWORKING
- CHAPTER 4: USING ADVANCED VSPHERE FEATURES

CHAPTER 1

Automating vCenter Server Deployment and Configuration

IN THIS CHAPTER, YOU WILL LEARN TO:

- **PREPARE THE VCENTER SERVER INSTALLATION** 4
- **CREATE AN AUTOMATED INSTALLATION** 5
- **SET UP YOUR VCENTER SERVER FOLDER STRUCTURE** 8
  - Creating a Folder Structure from Scratch 8
  - Exporting a Folder Structure 11
  - Importing a Folder Structure 14
- **DEFINE USERS AND THEIR PRIVILEGES** 15
  - Granting Privileges 15
  - Creating New Roles 19
  - Bringing in Users 21
  - Exporting Permissions 22
  - Importing Permissions 24
- **CONFIGURE DATACENTERS AND CLUSTERS** 26
  - Creating Datacenters 27
  - Creating Clusters 27
  - Configuring High Availability 27
  - Configuring Distributed Resource Scheduler 28
  - Configuring Enhanced vMotion Compatibility 29
  - Configuring Distributed Power Management 29
- **LICENSING** 31
  - Viewing License Information 31
  - Licensing a Host 33

CHAPTER 1

One of the focal points and key use cases of PowerCLI is the automation of tasks that are needed either as part of a disaster recovery (DR) solution or as part of an automated deployment solution that can be used repeatedly—You will be safe in the knowledge that the script will produce a consistent and easy-to-use solution.

This chapter will take you through some common areas automated within vSphere, starting at the beginning of the virtual infrastructure. Not only will we show you how to automate the build, but we'll also provide examples of export scripts that will help you export information into a centralized area, the exported data will then be ready for use in reports or for the import process of another setup.

Prepare the vCenter Installation

As part of the overall virtual infrastructure, one of the first areas you will need to install is the vCenter Server, or Virtual Infrastructure Server. Although this cannot be done directly using PowerCLI cmdlets, you can use the automated nature of PowerCLI and PowerShell to automate the install of vCenter Server.

The key thing to remember while reading this chapter—and indeed the entire book—is that PowerShell reaches past the virtual infrastructure. It can be used to manage most areas of the Windows-based operating system and application set. PowerCLI is purely an addition to PowerShell (known as a snap-in) that allows you to manage the virtual infrastructure.

To automate the installation of vCenter Server and its respective components, including the vSphere Client, Update Manager, Converter, and the corresponding databases, you will need the install media as well as various other items, such as the correct version of the .NET Framework and Windows installed on the server. The components you choose to install will depend on your infrastructure and the type of database you are going to use with your vCenter Server install.

Before you attempt to create an automated installation, be sure that

- The server meets at least the minimum hardware requirements as specified in the VMware ESX and vCenter Server installation documents provided by VMware.
- The server is configured with a static IP address.
- The computer name consists of fewer than 15 characters. (To conform to best practice, ensure that the computer name matches the hostname in the fully qualified domain name [FQDN] of the system.)

- The system is joined to a domain, and not a workgroup. While this is not a strict requirement, domain membership ensures that when you're using advanced features like the vCenter Guided Consolidation Service, the vCenter Server will be able to find all domains and systems on the network for the purpose of converting physical systems to virtual machines (VMs).
- A supported database is already available, unless you're using the bundled SQL Server 2005 Express Edition.
- A valid system data source name (DSN) exists that allows vCenter Server to connect to the created database.
- The vCenter Server is able to directly access the hosts it will manage without any network address translation between the server and the hosts.

NO MAGIC WANDS

Notice that all these requirements and recommendations are the same as those you'd check if you were manually installing vCenter Server on a single machine. People often think that scripting introduces some kind of magic or new ways to do things behind the scene. Not so! We use exactly the same methods VMware does for a manual install; it's just automated. If things go wrong, troubleshoot them the same way you would for a standard vCenter Server install that went wrong.

Create an Automated Installation

When installing vCenter Server manually, you first download the media and then run through a series of wizards, ensuring each step within the wizard is correctly configured before completing the installation and waiting for the wizard to install vCenter Server. This process can become cumbersome if the installation needs to be repeated multiple times, and mistakes can be made that could cause key configured items to be incorrect.

Use the script in Listing 1.1 as an example; it shows how you might automate the installation of vCenter Server while ensuring all database components are installed and all connections to the database are created. This example connects to a SQL 2005 database that was set up previously and is ready for install. With all

items clearly defined within the script, using a script like this ensures each installation is configured correctly and no mistakes are made.

LISTING 1.1 Sample script for an automated installation of vCenter Server

```
Function New-RegKey ($RegLocation, $RegKey, $RegValue) {
   If (Test-Path $RegLocation) {
   } Else {
      Write "Creating Registry Key $RegLocation"
      Mkdir $RegLocation | Out-Null
   }
   If (Get-ItemProperty $RegLocation $RegKey `
   -ErrorAction SilentlyContinue) {
       Write "Registry Key '$RegKey' already Exists."
   } Else {
       Write "Creating $RegKey with a value of $RegValue"
       New-ItemProperty -Path $RegLocation -Name $RegKey `
       -Value $RegValue `
       | Out-Null
   }
}

#Install VC unattended
$VCMedia = "C:\Temp\InstallMedia"
$LiKey = "XXX-XXX-XXX-XXX"
$Username = "My Name"
$CompanyName = "My Company"
$ODBCName = "vCenter Database"
$DBSrv = "SQL2005DB"
$DBUser = "VMware"
$DBPass = "VCDataba53"

# For SQL 2008 connections ensure the database client is installed
If (-Not (Test-Path 'C:\WINDOWS\system32\sqlncli10.dll')) {
 Write "SQL 2008 Native Client not found.
 Install it & then re-run this script"
 Exit
}

#Create DSN connection
```

```
$DrvPath = "C:\WINDOWS\system32\sqlncli10.dll"
New-Regkey "HKLM:SOFTWARE\ODBC\ODBC.INI\$ODBCName" `
"Driver" $DrvPath
New-RegKey "HKLM:SOFTWARE\ODBC\ODBC.INI\$ODBCName" `
"Description" $ODBCName
New-RegKey "HKLM:SOFTWARE\ODBC\ODBC.INI\$ODBCName" `
"Server" $DBSrv
New-RegKey "HKLM:SOFTWARE\ODBC\ODBC.INI\$ODBCName" `
"LastUser" $DBUser
New-RegKey "HKLM:SOFTWARE\ODBC\ODBC.INI\ODBC Data Sources" `
$ODBCName "SQL Server Native Client 10.0"

# Install vCenter
Write-Host "Installing vCenter"
$exe = "$VCmedia\vpx\VMware-vcserver.exe"
$args = '/q /s /w /L1033 /v" /qr USERNAME=\"$($username)\" '
$args = $agrs + `
'COMPANYNAME=\"$($Companyname)\" LICENSEKEY=\"$($LIKey)\" '
$args = $args + `
'DB_SERVER_TYPE=Custom DB_DSN=\"$($ODBCName)\" '
$args = $args + `
'DB_USERNAME=\"$($DBUser)\" DB_PASSWORD=\"$($DBPass)\" '
$args = $args + 'REBOOT=SUPPRESS'
Start-process $exe $args -Wait

# Initiate the Database Tables
Write-Host "DB tables"
$Exec = "$ENV:PROGRAMFILES\VMware\Infrastructure\VirtualCenter
Server\vpxd.exe"
Start-Process $Exec "-b" -Wait
Start-Service vpxd
```

Additional components, such as the vCenter Client or Update Manager, can easily be added to the previous script. Simply add a few extra lines in the install script, much like the vCenter Client install code that follows:

```
# Install vCenter Client
Write-Host "Installing vCenter Client"
Invoke-Item "$VCMedia\vpx\VMware-viclient.exe /s /w /v /qn `
/L*v %TEMP%\vmvcc.log `
WARNING_LEVEL=0"
```

To add the Host Update Utility component, try this next code:

```
# Install vCenter Client with Host Update Utility
Write-Host "Installing vCenter Client with Host Update Utility"
Invoke-Item "$VCMedia\vpx\VMware-viclient.exe /s /w /v /qn `
/L*v %TEMP%\vmvcc.log `
WARNING_LEVEL=0 `
INSTALL_VIUPDATE=1"
```

VMware supports more automated installation options and parameters, such as installing a linked mode vCenter Server, and maintains an online installation document here:

`http://www.vmware.com/pdf/vsp_4_vcserver_cmdline_install.pdf`

Set Up Your vCenter Server Folder Structure

Two types of folders are supported in vSphere. From within the Hosts and Clusters view, you are able to create folders at any point under the Datacenter level. These are commonly known as *yellow folders* and can be used throughout the infrastructure to organize the clusters, hosts, and VMs in a logical view.

Blue folders can be seen in the VMs and Templates view. Use these folders to more accurately reflect the layout of your VMs from a logical point of view. For example, you can create folders based on departments (such as Finance, Legal, and Customer Services) or by function (Internet, Active Directory, File Servers, Print Servers, Databases), or any other view that makes sense to your organization. Blue folders could also be used to reflect a security function and used to group the VMs into folders which only certain people can access. Once you create the folder, you can use it to grant access to various vCenter Server permissions.

Creating a Folder Structure from Scratch

You can initially create your folder structure when you create your new VMs; create your templates and move them into the appropriate folder. Another way of creating the folder structure is to plan the layout in a comma-separated variable (CSV) file. This type of plan can easily be created in an Excel document, as shown in Figure 1.1, and then exported to the CSV format needed to create the virtual folder structure.

FIGURE 1.1 Sample CSV layout

	A	B
1	Name	Path
2	Discovered virtual machine	vm\Discovered virtual machine
3	ESX Hosts	vm\ESX Hosts
4	Projects	vm\Projects
5	Templates	vm\Templates
6	Test VMs	vm\Test VMs
7	VDI	vm\VDI
8	vCenter	vm\vCenter
9	Windows 2008 Migration	vm\Projects\Windows 2008 Migration
10	Test	vm\Projects\Test
11	Developers Labs	vm\Projects\Developers Labs
12	Test VMs	vm\VDI\Test VMs

In the example CSV file we created, there are two columns. The first column, Name, is used to define the name of the folder that you wish to create. The second column, Path, is used to show the path to where this folder is to be created in vCenter Server. As seen in Figure 1.1, in the Path column all entries begin with `vm\`. This folder will not be created but is used by the underlying application programming interface (API). Once you have created the CSV file that contains the layout of your folder structure, a script can easily read your CSV file and create the structure using the code shown in Listing 1.2.

LISTING 1.2 Using a CSV file to create a vCenter file structure

```
function Import-Folders{
<#
.SYNOPSIS
  Imports a csv file of folders into vCenter Server and
  creates them automatically.
.DESCRIPTION
  The function will import folders from CSV file and create
  them in vCenter Server.
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER FolderType
  The type of folder to create
.PARAMETER DC
  The Datacenter to create the folder structure
.PARAMETER Filename
  The path of the CSV file to use when importing
```

```
.EXAMPLE 1
  PS> Import-Folders -FolderType "Blue" -DC "DC01" `
      -Filename "C:\BlueFolders.csv"
.EXAMPLE 2
  PS> Import-Folders -FolderType "Yellow" -DC "Datacenter"
  -Filename "C:\YellowFolders.csv"
#>

  param(
  [String]$FolderType,
  [String]$DC,
  [String]$Filename
  )

  process{
    $vmfolder = Import-Csv $filename | `
    Sort-Object -Property Path
   If ($FolderType -eq "Yellow") {
      $type = "host"
   } Else {
      $type = "vm"
   }
   foreach($folder in $VMfolder){
      $key = @()
      $key =  ($folder.Path -split "\\")[-2]
      if ($key -eq "vm") {
       get-datacenter $dc | get-folder $type | `
       New-Folder -Name $folder.Name
      } else {
        Get-Datacenter $dc | get-folder $type | `
        get-folder $key | `
            New-Folder -Name $folder.Name
      }
   }
  }
}

Import-Folders -FolderType "blue" -DC "DC01" `
-Filename "C:\BlueFolders.csv"
```

Exporting a Folder Structure

Both yellow and blue folder views can be exported to a CSV file. You will find this technique useful when you are rebuilding your vCenter Server from scratch or creating a DR replica of the current virtual infrastructure.

The script in Listing 1.3 can be used to export either a blue or a yellow folder structure to a CSV. It can also be used to export the location of the current VMs, ensuring a replicated location when you reimport the structure.

LISTING 1.3 Exporting a vCenter structure to a CSV file

```
Filter Get-FolderPath {
<#
.SYNOPSIS
  Colates the full folder path
.DESCRIPTION
  The function will find the full folder path returning a
  name and path
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
#>
    $_ | Get-View | % {
        $row = "" | select Name, Path
        $row.Name = $_.Name

        $current = Get-View $_.Parent
        $path = $_.Name
        do {
            $parent = $current
            if($parent.Name -ne "vm"){
             $path = $parent.Name + "\" + $path
            }
            $current = Get-View $current.Parent
        } while ($current.Parent -ne $null)
        $row.Path = $path
        $row
    }
}
```

```
Function Export-Folders {
  <#
.SYNOPSIS
  Creates a csv file of folders in vCenter Server.
.DESCRIPTION
  The function will export folders from vCenter Server
  and add them to a CSV file.
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER FolderType
  The type of folder to export
.PARAMETER DC
  The Datacenter where the folders reside
.PARAMETER Filename
  The path of the CSV file to use when exporting
.EXAMPLE 1
  PS> Export-Folders -FolderType "Blue" -DC "DC01" -Filename `
      "C:\BlueFolders.csv"
.EXAMPLE 2
  PS> Export-Folders -FolderType "Yellow" -DC "Datacenter"
  -Filename "C:\YellowFolders.csv"
#>

  param(
  [String]$FolderType,
  [String]$DC,
  [String]$Filename
  )

  Process {
   If ($Foldertype -eq "Yellow") {
      $type = "host"
   } Else {
     $type = "vm"
   }
   $report = @()
   $report = get-datacenter $dc | Get-folder $type | `
```

```
   get-folder | Get-Folderpath
   $Report | foreach {
    if ($type -eq "vm") {
     $_.Path = ($_.Path).Replace($dc + "\","$type\")
    }
   }
   $report | Export-Csv $filename -NoTypeInformation
  }
}

Function Export-VMLocation {
  <#
.SYNOPSIS
  Creates a csv file with the folder location of each VM.
.DESCRIPTION
  The function will export VM locations from vCenter Server
  and add them to a CSV file.
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER DC
  The Datacenter where the folders reside
.PARAMETER Filename
  The path of the CSV file to use when exporting
.EXAMPLE 1
  PS> Export-VMLocation -DC "DC01" `
      -Filename "C:\VMLocations.csv"
#>

  param(
  [String]$DC,
  [String]$Filename
  )

  Process {
   $report = @()
   $report = get-datacenter $dc | get-vm | Get-Folderpath
   $report | Export-Csv $filename -NoTypeInformation
```

```
  }
}

Export-Folders "Blue" "DC01" "C:\BlueFolders.csv"
Export-VMLocation "DC01" "C:\VMLocation.csv"
Export-Folders "Yellow" "DC01" "C:\YellowFolders.csv"
```

Importing a Folder Structure

You can import an existing blue or yellow folder structure into vCenter Server using the `Import-Folders` function previously shown in Listing 1.2. You can also choose if you would like your VMs moved back into their correct blue folders by using the `Import-VMLocation` function, as shown in Listing 1.4.

LISTING 1.4 Importing VMs to their blue folders

```
Function Import-VMLocation {
 <#
.SYNOPSIS
  Imports the VMs back into their Blue Folders based on
  the data from a csv file.
.DESCRIPTION
  The function will import VM locations from CSV File
  and add them to their correct Blue Folders.
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER DC
  The Datacenter where the folders reside
.PARAMETER Filename
  The path of the CSV file to use when importing
.EXAMPLE 1
  PS> Import-VMLocation -DC "DC01" -Filename "C:\VMLocations.csv"
#>

  param(
  [String]$DC,
  [String]$Filename
```

```
  )

  Process {
   $Report = @()
   $Report = import-csv $filename | Sort-Object -Property Path
   foreach($vmpath in $Report){
      $key = @()
      $key =  Split-Path $vmpath.Path | split-path -leaf
      Move-VM (get-datacenter $dc `
      | Get-VM $vmpath.Name) `
      -Destination (get-datacenter $dc | Get-folder $key)
   }
  }
}

Import-VMLocation "DC01" "C:\VMLocation.csv"
```

Define Users and Their Privileges

The authorization to perform tasks in your virtual infrastructure is controlled by a role-based access control (RBAC) system. A vCenter Server administrator can specify in great detail which users or groups can perform which tasks on which objects. RBAC systems are defined using three key concepts:

Privilege A privilege is the ability to perform an action or read a property. Examples include powering on a VM or adding a folder.

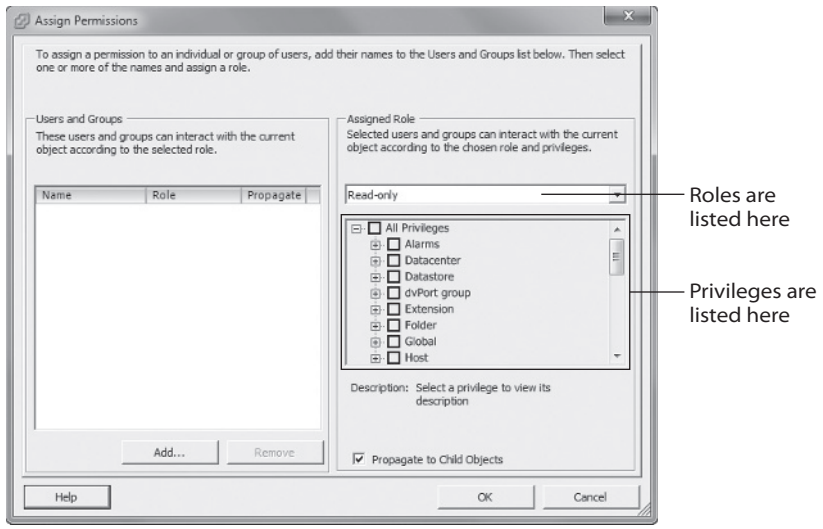
Role A role is a collection of privileges. Roles provide a way to add all the individual privileges that are required to perform a number of tasks, such as administer a vSphere host.

Object An object is an item on which actions can be performed. vCenter Server objects are datacenters, folders, resource pools, clusters, hosts, and VMs.

Granting Privileges

Privileges are found in the vSphere Client. When using the Assign Permissions wizard, you are able to add new permissions. The privileges are listed in Figure 1.2.

FIGURE 1.2 vCenter Server Privileges


Assign Permissions
To assign a permission to an individual or group of users, add their names to the Users and Groups list below. Then select one or more of the names and assign a role.
Users and Groups
These users and groups can interact with the current object according to the selected role.
Name
Role
Propagate
Add...
Remove
Assigned Role
Selected users and groups can interact with the current object according to the chosen role and privileges.
Read-only
All Privileges
Alarms
Datacenter
Datastore
dvPort group
Extension
Folder
Global
Host
Description: Select a privilege to view its description
Propagate to Child Objects
Help
OK
Cancel
Roles are listed here
Privileges are listed here

How many privileges are there? Think of any action you have ever performed in the vCenter Client. Think about the actions you have not yet come across or used in your everyday job. Now add them up, and you will have some idea of how many privileges there are in vCenter Server. Luckily, we are able to use PowerCLI to come up with a scientific answer for this question. You can easily list all privileges available to assign to a user through vCenter Server using the `Get-VIPrivilege` cmdlet.

```
[vSphere PowerCLI] C:\> Get-VIPrivilege | Select Name, Description

Name                     Description
----                     -----------
Anonymous                The only privilege held by sessions ...
View                     Visibility without read access to an...
Read                     Grants read access to an entity
Manage custom attributes Add, remove, and rename custom attri...
Set custom attribute     Set the value of a custom attribute ...
Log event                Log a user-defined event on an object
Cancel task              Cancel a running task
Licenses                 Manage licenses
Diagnostics              Export diagnostic data
Settings                 Edit global settings
Act as vCenter Server    Act as the vCenter Server
```

```
Capacity planning          Discover and convert physical host t...
Script action              Schedule an external script action
Proxy                      Add or remove endpoints to or from t...
Disable methods            Operations are disabled in vCenter
Enable methods             Operations are enabled in vCenter
Service managers           Access the directory service
Health                     Access the health of vCenter group
.....
```

We purposely truncated the output listing due to the large number of privileges available. You can count the number of privileges available for assigning to your roles and users or groups by using the `Measure-Object` cmdlet:

```
[vSphere PowerCLI] C:\> Get-VIPrivilege | Measure-Object

Count    : 266
Average  :
Sum      :
Maximum  :
Minimum  :
Property :
```

You can also use the `Get-VIPrivilege` cmdlet to show only the privileges available to certain sets of objects like a host:

```
[vSphere PowerCLI] C:\> Get-VIPrivilege -Name *Host*

Name                               Id
----                               --
Host operation                     DVSwitch.HostOp
Add standalone host                Host.Inventory.AddStandaloneHost
Add host to cluster                Host.Inventory.AddHostToCluster
Remove host                        Host.Inventory.
RemoveHostFromClu...
Move cluster or standalone host    Host.Inventory.MoveCluster
Move host                          Host.Inventory.MoveHost
Add host to vCenter                Host.Local.InstallAgent
Host USB device                    VirtualMachine.Config.
HostUSBDevice
Host                               Host
Host profile                       Profile
```

You can view which groups (collections of privileges) are available by using the `Get-VIPrivilege` cmdlet with the `-PrivilegeGroup` parameter, as shown here:

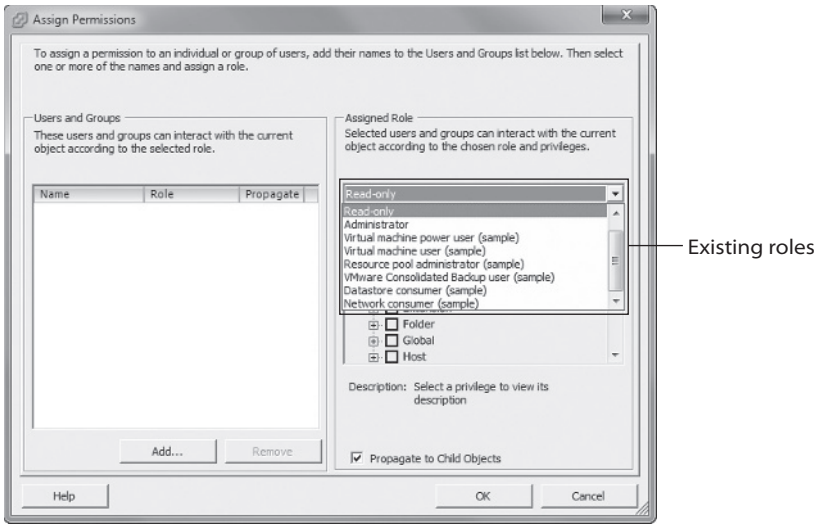
```
[vSphere PowerCLI] C:\> Get-VIPrivilege -PrivilegeGroup | `
                        Select Name, Description
Name                              Description
----                              -----------
System                            System
Global                            Global
Folder                            Folder
Datacenter                        Datacenter
Datastore                         Datastore
Network                           Networks
vNetwork Distributed Switch       vNetwork Distributed Switch
dvPort group                      dvPort groups
Host                              Host
Inventory                         Host inventory
Configuration                     Host configuration
Local operations                  Host local operations
CIM                               CIM
Virtual machine                   Virtual machine
Inventory                         Virtual machine inventory
Interaction                       Virtual machine interaction
Configuration                     Virtual machine configuration
State                             Virtual machine state
Provisioning                      Virtual machine provisioning
VRMPolicy                         Virtual Rights Management Policy
Resource                          Resource allocation
Alarms                            Alarms
Tasks                             Tasks
Scheduled task                    Scheduled task
Sessions                          Sessions
Performance                       Performance
Permissions                       Permissions
Extension                         Extensions
vApp                              Privileges related to vApps
Host profile                      Host profile
Storage views                     Storage views
VMware vCenter Update Manager     VMware vCenter Update Manager
```

```
Manage Baseline                  Manage baselines
Upload file                      Upload file
Configure                        General VMware vCenter Upd...
Manage Patches and Upgrades      Manage virtual machine and...
```

Creating New Roles

Roles can be found in the vSphere Client whenever you add a new permission. The Assigned Role drop-down box in the Assign Permissions dialog box shown in Figure 1.3 lists your existing roles.

FIGURE 1.3 vCenter Server roles


Assign Permissions
To assign a permission to an individual or group of users, add their names to the Users and Groups list below. Then select one or more of the names and assign a role.
Users and Groups
These users and groups can interact with the current object according to the selected role.
Name
Role
Propagate
Add...
Remove
Assigned Role
Selected users and groups can interact with the current object according to the chosen role and privileges.
Read-only
Read-only
Administrator
Virtual machine power user (sample)
Virtual machine user (sample)
Resource pool administrator (sample)
VMware Consolidated Backup user (sample)
Datastore consumer (sample)
Network consumer (sample)
Folder
Global
Host
Description: Select a privilege to view its description
Propagate to Child Objects
Help
OK
Cancel
Existing roles

You can see an overview of the predefined roles by using the `Get-VIRole` cmdlet, as shown here:

```
[vSphere PowerCLI] C:\> Get-VIRole | Select Name, Description

Name                             Description
----                             -----------
NoAccess                         Used for restricting granted access
Anonymous                        Not logged-in user (cannot be granted)
View                             Visibility access (cannot be granted)
ReadOnly                         See details of objects, but not make...
Admin                            Full access rights
```

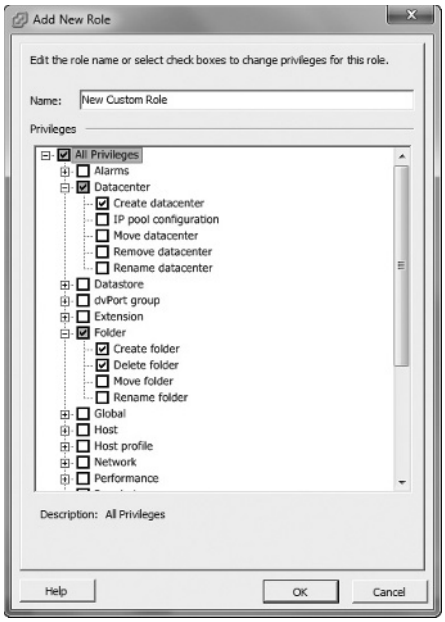
```
VirtualMachinePowerUser      Provides virtual machine interaction...
VirtualMachineUser           Provides virtual machine interaction...
ResourcePoolAdministrator    Supports delegated resource management
VMwareConsolidatedBackupUser Used by the Consolidated Backup utility
DatastoreConsumer            Assigned to datastores to allow crea...
NetworkConsumer              Assigned to networks to allow associ...
```

Now that you know that a role is a group of privileges and you've learned to use the `Get-VIPrivilege` and `Get-VIRole` cmdlets, we want to introduce you to `New-VIRole`. You can use the `New-VIRole` cmdlet with `Get-VIPrivilege` to define a new role. You can define your own group of privileges, which can later be assigned to your users. An example is shown in Listing 1.5; you can see the results in the vCenter Client, as shown in Figure 1.4.

LISTING 1.5 Creating a new role

```
New-VIRole `
-Name 'New Custom Role' `
-Privilege (Get-VIPrivilege `
-PrivilegeGroup "Interaction","Provisioning")
```

FIGURE 1.4 New roles



A new role can also be created at a granular level. First choose the privileges you want to use:

```
$Priv = @()
$MyPriv = "Profile", "VCIntegrity.Baseline", `
"VApp.Move", "Profile.Clear"
```

And then add each of them into a custom object:

```
Foreach ($CustPriv in $MyPriv){
   $Priv += Get-VIPrivilege | Where {$_.Id -eq $CustPriv}
}
```

You can then use the custom object to apply your specific permissions to the new role:

```
New-VIRole "New selected Role" -Privilege $Priv
```

Bringing in Users

Now that you have defined your roles, you can start using them. Until now, you have only been working with roles and privileges. Once you define what you want your user to be able to do, you need to add users and grant them access to the roles. You can then enable them to start using the features of the vSphere Client.

A role or privilege can be assigned to any of the objects within a vCenter Server. Each of the objects can be defined by different roles or privileges. Together, objects, roles, and privileges make up a *permission set*. Permission sets can be inherited; inheritance ensures that each object underneath a datacenter, cluster, resource pool, or folder gives the users the correct access privileges.

So it comes as no great surprise that, when adding a permission through PowerCLI, you must consider three areas:

Role The role which you will assign to the user

Principal The user or group to which you wish to assign permissions

Entity The object, folder, cluster, datacenter, or resource pool for which you would like to grant permissions to the user

In the code that follows, we grant a user (`MyDomain\User01`) access to `New Custom Role` at the datacenter level:

```
New-VIPermission -Role 'New Custom Role' `
-Principal 'MYDOMAIN\User01' `
-Entity (Get-Datacenter)
```

After you've set up and tested individual permissions, you can export them to a readable, importable format. This eases multiple installations and the transfer of permissions to further vCenter Servers, and ensures consistency as well. We'll show you how next.

Exporting Permissions

The script in Listing 1.6 exports all relevant information into a CSV file, which can later be used to import them back into the same or a different vCenter Server. Exporting the permissions can be a great way to satisfy a security audit or ensure the relevant departments or users have the correct permissions.

LISTING 1.6 Exporting permissions

```
Function Export-PermissionsToCSV {
 <#
.SYNOPSIS
  Exports all Permissions to CSV file
.DESCRIPTION
  The function will export all permissions to a CSV
  based file for later import
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER Filename
  The path of the CSV file to be created
.EXAMPLE 1
  PS> Export-PermissionsToCSV -Filename "C:\Temp\Permissions.csv"
#>


  param(
  [String]$Filename
  )

  Process {
   $folderperms = get-datacenter | Get-Folder | Get-VIPermission
   $vmperms = Get-Datacenter | get-vm | Get-VIPermission


   $permissions = get-datacenter | Get-VIpermission
```

```
$report = @()
   foreach($perm in $permissions){
     $row = "" | select EntityId, Name, Role, `
     Principal, IsGroup, Propagate
     $row.EntityId = $perm.EntityId
     $Foldername = (Get-View -id $perm.EntityId).Name
     $row.Name = $foldername
     $row.Principal = $perm.Principal
     $row.Role = $perm.Role
     $row.IsGroup = $perm.IsGroup
     $row.Propagate = $perm.Propagate
     $report += $row
}


foreach($perm in $folderperms){
     $row = "" | select EntityId, Name, Role, `
     Principal, IsGroup, Propagate
     $row.EntityId = $perm.EntityId
     $Foldername = (Get-View -id $perm.EntityId).Name
     $row.Name = $foldername
     $row.Principal = $perm.Principal
     $row.Role = $perm.Role
     $row.IsGroup = $perm.IsGroup
     $row.Propagate = $perm.Propagate
     $report += $row
}


foreach($perm in $vmperms){
     $row = "" | select EntityId, Name, Role, `
     Principal, IsGroup, Propagate
     $row.EntityId = $perm.EntityId
     $Foldername = (Get-View -id $perm.EntityId).Name
     $row.Name = $foldername
     $row.Principal = $perm.Principal
     $row.Role = $perm.Role
     $row.IsGroup = $perm.IsGroup
     $row.Propagate = $perm.Propagate
     $report += $row
}
```

```
    $report | export-csv $Filename -NoTypeInformation
  }
}

Export-PermissionsToCSV -Filename "C:\Temp\Permissions.csv"
```

Importing Permissions

It is equally important to be able to import the permissions back into your vCenter Server. You can use the script in Listing 1.7.

LISTING 1.7 Importing permissions

```
function Import-Permissions {
<#
.SYNOPSIS
  Imports all Permissions from CSV file
.DESCRIPTION
  The function will import all permissions from a CSV
  file and apply them to the vCenter Server objects.
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER DC
  The Datacenter to import the permissions into
.PARAMETER Filename
  The path of the CSV file to be imported
.EXAMPLE 1
  PS> Import-Permissions -DC "DC01" `
      -Filename "C:\Temp\Permissions.csv"
#>


param(
[String]$DC,
[String]$Filename
)


process {
```

```
$permissions = @()
$permissions = Import-Csv $Filename
foreach ($perm in $permissions) {
 $entity = ""
 $entity = New-Object VMware.Vim.ManagedObjectReference
 $object = Get-Inventory -Name $perm.Name
 if($object.Count){
  $object = $object | where {$_.Id -eq $perm.EntityId}
 }
 if($object){
  switch -wildcard ($perm.EntityId)
  {
   Folder* {
    $entity.type = "Folder"
    $entity.value = $object.Id.Trimstart("Folder-")
   }
   VirtualMachine* {
    $entity.Type = "VirtualMachine"
    $entity.value = $object.Id.Trimstart("VirtualMachine-")
   }
   ClusterComputeResource* {
    $entity.Type = "ClusterComputeResource"
    $entity.value = `
    $object.Id.Trimstart("ClusterComputeResource-")
   }
   Datacenter* {
    $entity.Type = "Datacenter"
    $entity.value = $object.Id.Trimstart("Datacenter-")
   }
  }
  $setperm = New-Object VMware.Vim.Permission
  $setperm.principal = $perm.Principal
  if ($perm.isgroup -eq "True") {
   $setperm.group = $true
  } else {
   $setperm.group = $false
  }
  $setperm.roleId = (Get-virole $perm.Role).id
  if ($perm.propagate -eq "True") {
   $setperm.propagate = $true
```

```
    } else {
     $setperm.propagate = $false
    }
    $doactual = Get-View -Id `
    'AuthorizationManager-AuthorizationManager'
    Write-Host "Setting Permissions on `
    $($perm.Name) for $($perm.principal)"
    $doactual.SetEntityPermissions($entity, $setperm)
   }
  }
  }
}


Import-Permissions -DC "DC01" -Filename "C:\Temp\Permissions.csv"
```

Configure Datacenters and Clusters

vCenter Server has a hierarchical management structure similar to that of Microsoft Active Directory. Three main containers can be added to vCenter Server:

- Datacenters
- Clusters
- Folders

Datacenters A datacenter is a logical container within vCenter Server used to store clusters, folders, and VMs; these are often named for the physical location where the hosts reside, such as "Boston" or "South West Datacenter."

Clusters A cluster is defined as a group of like-configured computers that act in a fully redundant setup to ensure availability of applications and operating systems. A vCenter Server cluster is no different. Clusters are used in vCenter Server for three main functions: high availability, load balancing, and high-performance computing. A cluster is made up of two or more physical servers that provide resources for the hosts that are assigned to that cluster.

Folders A folder is a logical way to define how VMs or other vCenter Server objects are organized. Folders are often used to organize VMs into department owners or server functions.

Creating Datacenters

Datacenters are generally created as part of the initial setup process. The setup can be automated by using the following code, which will create a datacenter called Boston and store it in a variable. The `Datacenter` object held within the variable can then be referred to later in the code as you create clusters or folders:

```
$BostonDC = New-Datacenter -Name Boston
```

Creating Clusters

Clusters are more complex than datacenters; there are many configurable items available for a new cluster. Consider the options the vSphere Client gives us: the normal cluster options as well as configuration options for VMware High Availability (HA), VMware Distributed Resource Scheduler (DRS), VMware Enhanced VMotion Compatibility (EVC), and VMware Distributed Power Management (DPM).

To create a new cluster in the Boston datacenter you created earlier, you can use the following code:

```
$ProductionCluster = New-Cluster -Location $BostonDC `
-Name "Production"
```

This code line gives you the basic settings. The sections that follow discuss the additional cluster settings available to you.

Configuring High Availability

When configured in a cluster, VMWare HA gives you many advantages, including the following:

- Proactive monitoring of all vSphere hosts and VMs
- Automatic detection of vSphere host failure
- Rapid restart of VMs affected by host failure
- Optimal placement of VMs after server failure

Much like the configuration of a cluster through the vSphere client, you can configure HA within a cluster either as part of the initial cluster setup or you can alter an existing cluster object. For example, to configure a new cluster named `Production` with HA enabled and an HA failover level of 1 physical host failure and the HA Restart Priority as Medium, you would use the code in Listing 1.8.

LISTING 1.8 Enabling HA with a failover host level and Restart Priority on a new cluster

```
$ProductionCluster = New-Cluster `
-Location $BostonDC `
-Name "Production" `
-HAEnabled -HAAdmissionControlEnabled `
-HAFailoverLevel 1 `
-HARestartPriority "Medium"
```

To complete this same action on an existing cluster, you first need to retrieve the cluster as an object and then push it down the pipeline into the `Set-Cluster` cmdlet, as shown in Listing 1.9.

LISTING 1.9 Enabling HA with a failover host level and restart priority on an existing cluster

```
Get-Cluster `
-Location $BostonDC `
-Name "Production" | `
Set-Cluster -HAEnabled $true `
-HAAdmissionControlEnabled $true `
-HAFailoverLevel 1 `
-HARestartPriority "Medium"
```

Configuring Distributed Resource Scheduler

VMware DRS is a configuration made at the cluster level of the vCenter Server environment that balances VM workloads with available host resources. With VMware DRS, you are able to define the rules for allocation of physical resources among the VMs. DRS can be configured for manual or automatic control. If the workload on one or more VMs drastically changes, DRS redistributes the VMs among the physical servers to ensure the resources are available where needed. Much like HA, DRS can be configured as part of the initial cluster setup or as an alteration to an existing cluster object. For example, to configure a new `Production` cluster with DRS enabled and a DRS automation level of `FullyAutomated` with `DRSMode` set to `FullyAutomated`, you would use the code in Listing 1.10.

LISTING 1.10 Configuring DRS on a new cluster

```
$ProductionCluster = New-Cluster "Production" `
-DrsEnabled `
-DrsAutomationLevel "FullyAutomated" `
-DrsMode "FullyAutomated"
```

To complete this same action on an existing cluster, you would again need to retrieve the cluster object and push the object through the pipe into the `Set-Cluster` cmdlet, as shown in Listing 1.11.

LISTING 1.11 Configuring DRS on an existing cluster

```
Get-Cluster -Location $BostonDC `
-Name "Production" | Set-Cluster `
-DrsEnabled $true `
-DrsAutomationLevel "FullyAutomated" `
-DrsMode "FullyAutomated"
```

Configuring Enhanced vMotion Compatibility

EVC allows you to add multiple hosts with different CPU architectures to your cluster. EVC will, for example, allow you to add older hosts with Intel processors to a cluster that includes hosts with newer Intel processors. It does this by setting a mask on the VMs and ensuring the instruction sets are the same for both sets of hosts. Unfortunately, at this point in time VMware does not include either a PowerCLI cmdlet or a method to enable this feature programmatically. Therefore, configuring EVC is outside the scope of this book.

Configuring Distributed Power Management

DPM provides cost savings by dynamically consolidating VMs onto fewer hosts during periods of low usage. Once the VMs are consolidated onto fewer hosts, the remaining hosts that are no longer hosting any VMs are powered off to save power. Once utilization starts to increase, the vSphere Server will power these hosts back on as needed.

While there are currently no options to enable DPM through the native cmdlets that are provided with PowerCLI, you can address the API and create your own function to enable DPM. For more information about using the SDK/API or Project Onyx, read Chapter 19, "Onyx and the SDK."

Listing 1.12 shows how you can enable DPM on a cluster.

LISTING 1.12 Configuring DPM on a cluster

```
Function Set-DPM {
 <#
.SYNOPSIS
  Enables Distributed Power Management on a cluster
```

```
.DESCRIPTION
  This function will allow you to configure
  DPM on an existing vCenter Server cluster
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER Cluster
  The cluster on which to set DPM configuration
.PARAMETER Behavior
  DPM Behavior, this can be set to "off", "manual"
  or "Automated", by default it is "off"
.EXAMPLE 1
  PS> Set-DPM -Cluster "Cluster01" -Behavior "Automated"
#>

param(
  [String]$Cluster,
  [String]$Behavior
  )

  Process {
   switch ($Behavior) {
            "Off" {
               $DPMBehavior = "Automated"
               $Enabled = $false
            }
            "Automated" {
               $DPMBehavior = "Automated"
               $Enabled = $true
            }
            "Manual" {
               $DPMBehavior = "Manual"
               $Enabled = $true
            }
            default {
               $DPMBehavior = "Automated"
               $Enabled = $false
            }
      }
```

```
    $clus = Get-Cluster $Cluster | Get-View
    $spec = New-Object vmware.Vim.ClusterConfigSpecEx
    $spec.dpmConfig = New-Object VMware.Vim.ClusterDpmConfigInfo
    $spec.DpmConfig.DefaultDpmBehavior = $DPMBehavior
    $spec.DpmConfig.Enabled = $Enabled
    $clus.ReconfigureComputeResource_Task($spec, $true)
   }
}

Set-DPM -Cluster "Cluster01" -Behavior "Automated"
```

Licensing

Licensing is one of the first areas that will be critical to setting up a new host. Without a valid license, you can manage and use your host-to-host VMs for only 60 days.

You may be surprised to learn that there are no cmdlets to help with licensing ESX hosts or even viewing the current license details. However, the licensing information is available through the `Get-View` cmdlet, and you can manipulate the SDK to perform the actions necessary to both view license information and set the license key for your hosts. You can write functions to help you deal with these cmdlets and make them a little friendlier than the SDK code.

Viewing License Information

To make things easier, you can use the functions we'll show you next to list all license keys registered on the vCenter Server and also to set a license key on a host. The `Get-LicenseKey` function in Listing 1.13 lists all existing license keys.

LISTING 1.13 Retrieving license key information from vCenter Server

```
Function Get-LicenseKey {
 <#
.SYNOPSIS
  Retrieves License Key information
.DESCRIPTION
  This function will list all license keys added to
  vCenter Server
```

```
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.EXAMPLE 1
  PS> Get-LicenseKey
#>

  Process {
   $servInst = Get-View ServiceInstance
   $licMgr = Get-View `
   (Get-View ServiceInstance).Content.licenseManager
   $licMgr.Licenses
  }
}

Get-LicenseKey
```

Each of the existing license keys will be returned in an output listing like this:

```
  LicenseKey      : 00000-00000-00000-00000-00000
  EditionKey      : eval
  Name            : Product Evaluation
  Total           : 0
  Used            : 0
  CostUnit        :
  Properties      :
  Labels          :
  DynamicType     :
  DynamicProperty :

  LicenseKey      : AAAAA-BBBBB-CCCCC-DDDDD-EEEEE
  EditionKey      : esxEnterprisePlus
  Name            : vSphere 4 Enterprise Plus
  Total           : 0
  Used            : 2
  CostUnit        : cpuPackage:12core
  Properties      : {ProductName, ProductVersion, feature...}
  Labels          :
  DynamicType     :
```

```
DynamicProperty :

LicenseKey      : AAAAA-BBBBB-CCCCC-DDDDD-EEEEE
EditionKey      : vc
Name            : vCenter Server 4 Standard
Total           : 0
Used            : 1
CostUnit        : server
Properties      : {ProductName, ProductVersion, feature...}
Labels          :
DynamicType     :
DynamicProperty :
```

Licensing a Host

Once you have a list of the keys, you can use that information to license the ESX hosts attached to the vCenter Server. Listing 1.14 shows how you set the license key for a specific host.

LISTING 1.14 Adding a license key to a host

```
Function Set-LicenseKey {
 <#
.SYNOPSIS
  Sets a License Key for a host
.DESCRIPTION
  This function will set a license key for a host
  which is attached to a vCenter Server
.NOTES
  Source:  Automating vSphere Administration
  Authors: Luc Dekens, Arnim van Lieshout, Jonathan Medd,
           Alan Renouf, Glenn Sizemore
.PARAMETER LicKey
  The License Key
.PARAMETER VMHost
  The vSphere host to add the license key to
.PARAMETER Name
  The friendly name to give the license key
.EXAMPLE 1
  PS> Set-LicenseKey -LicKey "AAAAA-BBBBB-CCCCC-DDDDD-EEEEE" `
```

```
        -VMHost "esxhost01.mydomain.com" `
        -Name $null
#>

param(
  [String]$VMHost,
  [String]$LicKey,
  [String]$Name
  )

  Process {
   $vmhostId = (Get-VMHost $VMHost | Get-View).Config.Host.Value
   $servInst = Get-View ServiceInstance
   $licMgr = Get-View $servInst.Content.licenseManager
   $licAssignMgr = Get-View $licMgr.licenseAssignmentManager

   $license = New-Object VMware.Vim.LicenseManagerLicenseInfo
   $license.LicenseKey = $LicKey
   $licAssignMgr.UpdateAssignedLicense(`
   $VMHostId, $license.LicenseKey, $Name)
   }
}

Set-LicenseKey -LicKey "AAAAA-BBBBB-CCCCC-DDDDD-EEEEE" `
-VMHost "esxhost01.mydomain.com" `
-Name $null
```